AE

(12) UK Patent Application (19) GB (11) 2 349 242 (13) A

(43) Date of A Publication 25.10.2000

(21) Application No 9908869.2

(22) Date of Filing 20.04.1999

(71) Applicant(s)
Inventec Corporation
(Incorporated in Taiwan)
No 66 Hou-Kang St, Shih-Lin District, Taipei, Taiwan

(72) Inventor(s)
Jeffrey Lai
Kai Yang

(74) Agent and/or Address for Service
Lewis & Taylor
144 New Walk, LEICESTER, LE1 7JA, United Kingdom

(51) INT CL^7
G11C 16/02, G06F 12/10

(52) UK CL (Edition R)
G4A AMX ANV

(56) Documents Cited
GB 2333620 A GB 2291990 A WO 98/44420 A1
JP 090161491 A US 5682497 A US 5566314 A

(58) Field of Search
UK CL (Edition Q) G4A AFL AMX ANR ANV
INT CL^6 G06F 12/02 12/06 12/10, G11C 16/02 16/06
Online: WPI, EPODOC, INSPEC, COMPUTER

(54) Abstract Title
Flash memory architecture and rewrite method

(57) A flash memory architecture comprises master-slave FAT (file allocated table) and logical pages. The logical pages are defined subject to the number of erasable blocks at the flash memory (Fig. 2). When rewriting the flash memory, an index is set up corresponding to allocated blocks, and, whereas in the prior art, a whole file block would need to be erased before rewriting, according to the invention, the renewal content and the content in the original allocated block are combined and then written in a new allocated block. At the same time the index is modified and aimed at the new allocated block, enabling the new allocated block to substitute for the original allocated block, so as to eliminate the operation of erasing the whole file allocated block and the possibility of a file loss due to power failure, system down, or a replacement of the flash memory.

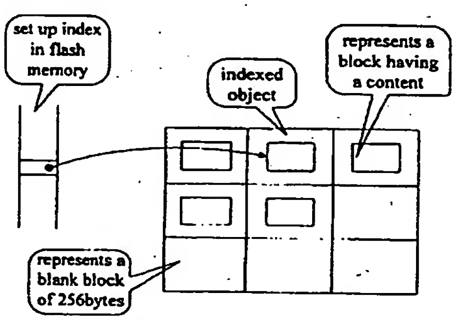

Fig. 1A

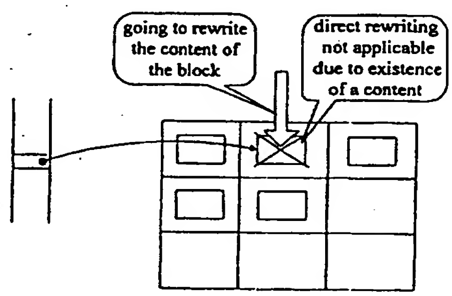

Fig. 1B

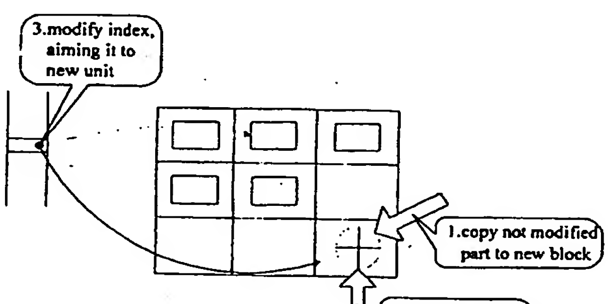

Fig. 1C

GB 2 349 242 A

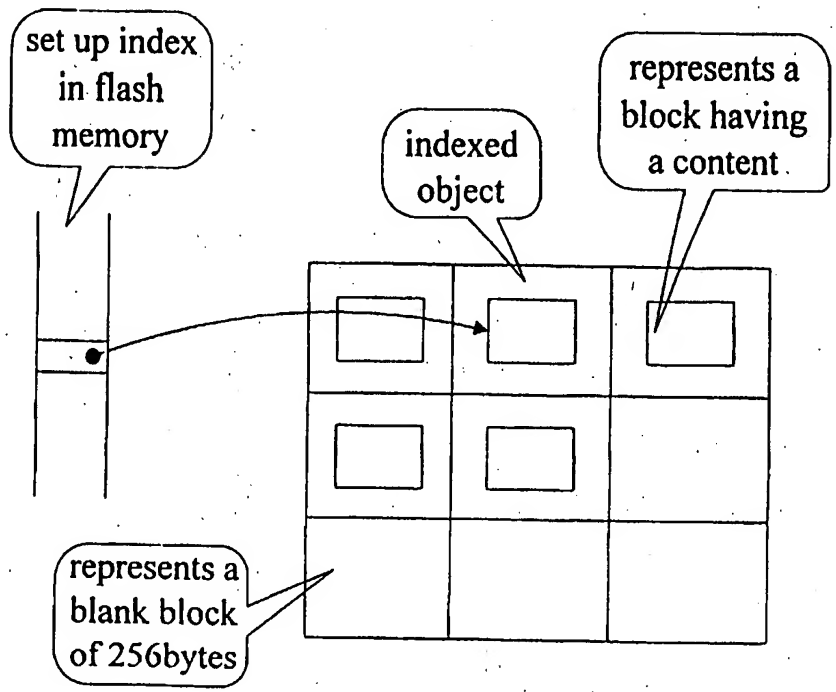
set up index in flash memory
indexed object
represents a block having a content
represents a blank block of 256bytes

Fig. 1A

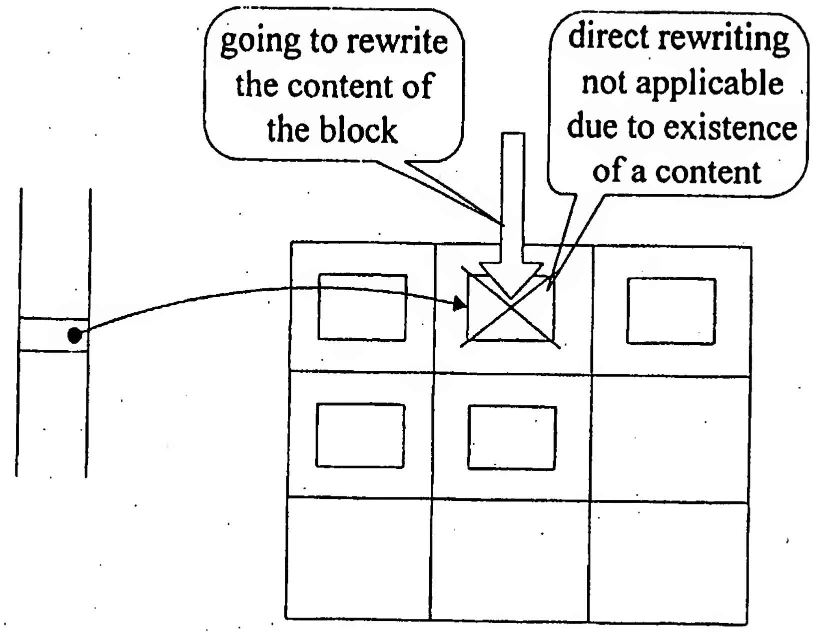
going to rewrite the content of the block
direct rewriting not applicable due to existence of a content

Fig. 1B

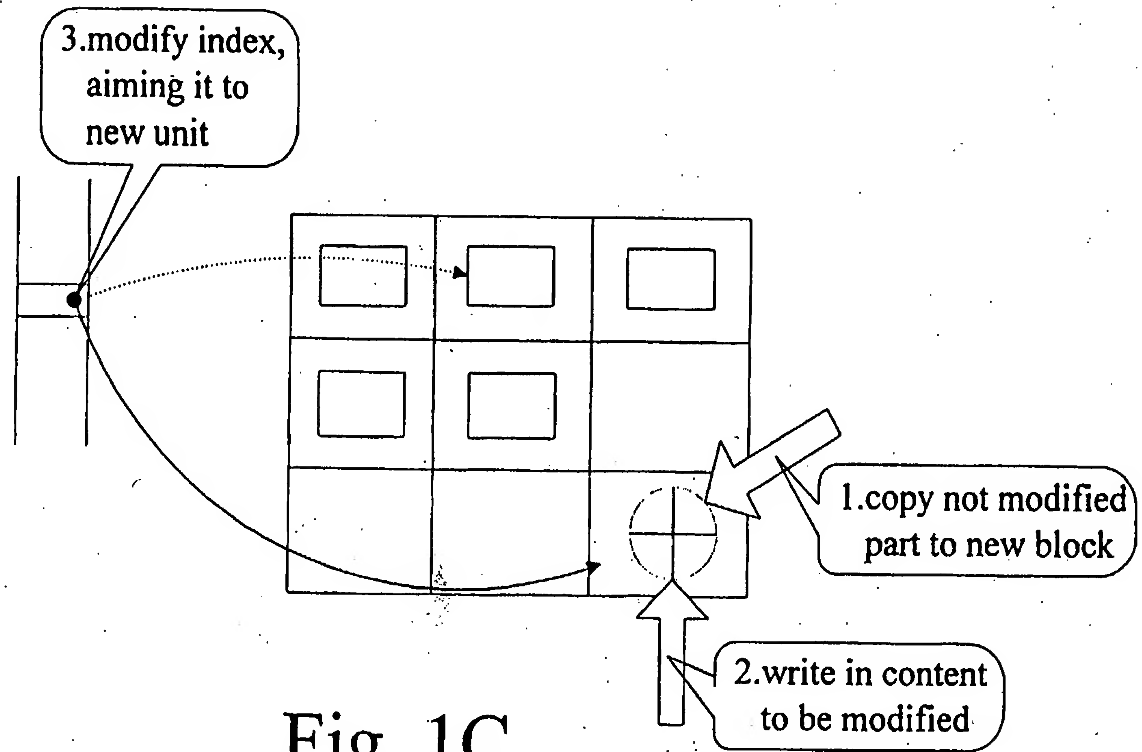
3.modify index, aiming it to new unit
1.copy not modified part to new block
2.write in content to be modified

Fig. 1C

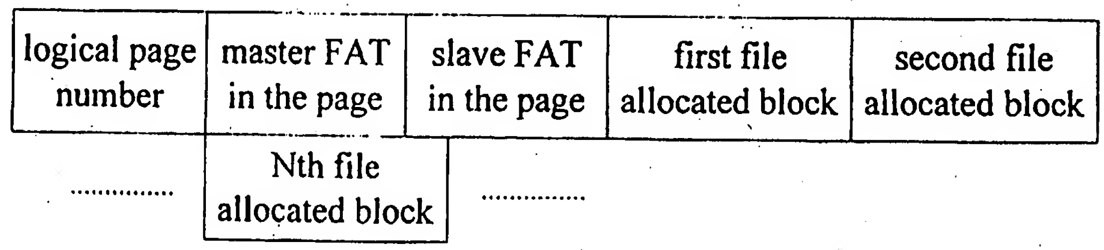
logical page number
master FAT in the page
slave FAT in the page
first file allocated block
second file allocated block
....
Nth file allocated block
....

Fig. 2A

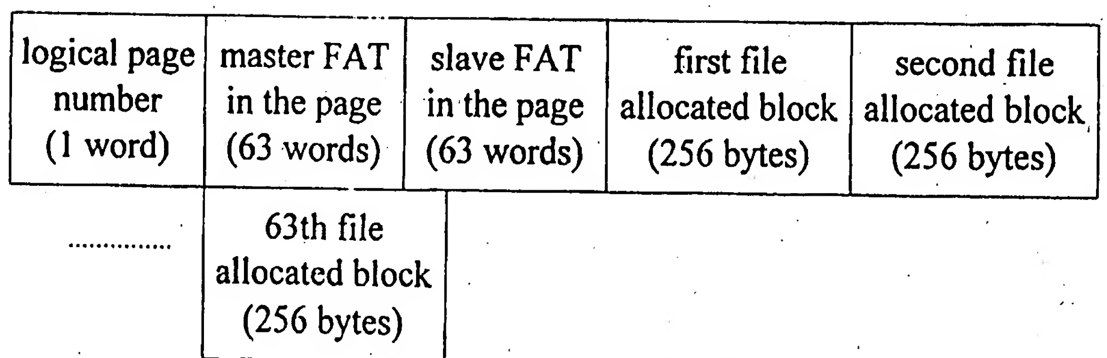
logical page number (1 word)
master FAT in the page (63 words)
slave FAT in the page (63 words)
first file allocated block (256 bytes)
second file allocated block (256 bytes)
....
63th file allocated block (256 bytes)

Fig. 2B

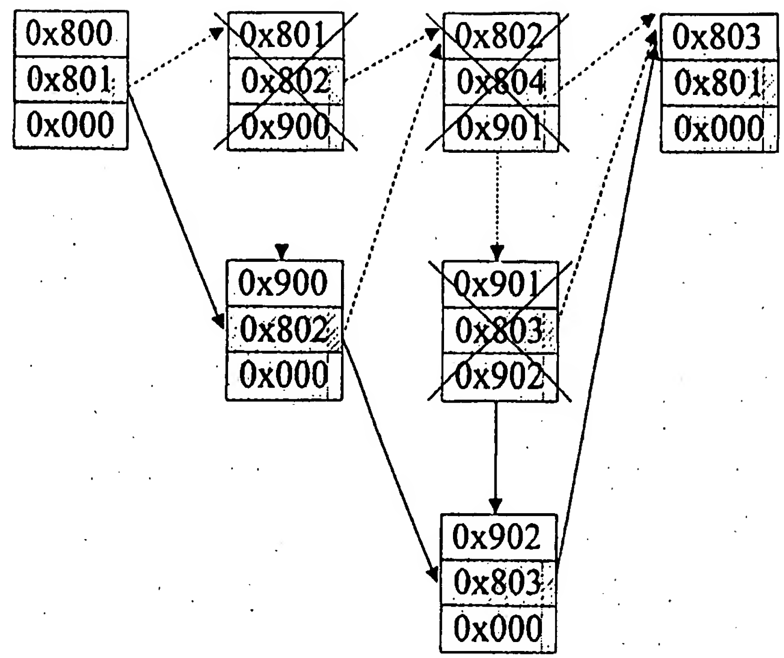
0x800
0x801
0x000
0x801
0x802
0x900
0x802
0x804
0x901
0x803
0x801
0x000
0x900
0x802
0x000
0x901
0x803
0x902
0x902
0x803
0x000

Fig. 3

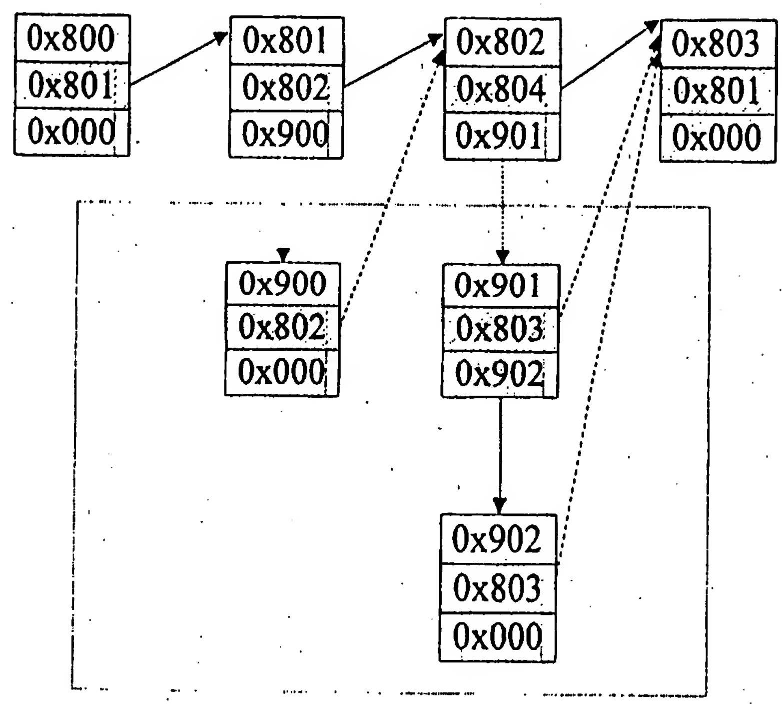
0x800
0x801
0x000
0x801
0x802
0x900
0x802
0x804
0x901
0x803
0x801
0x000
0x900
0x802
0x000
0x901
0x803
0x902
0x902
0x803
0x000

Fig. 4

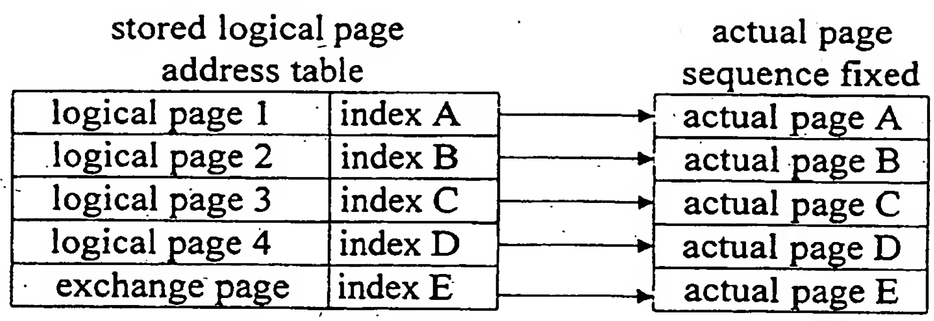
stored logical page address table
logical page 1
index A
logical page 2
index B
logical page 3
index C
logical page 4
index D
exchange page
index E
actual page sequence fixed
actual page A
actual page B
actual page C
actual page D
actual page E

Fig. 5A

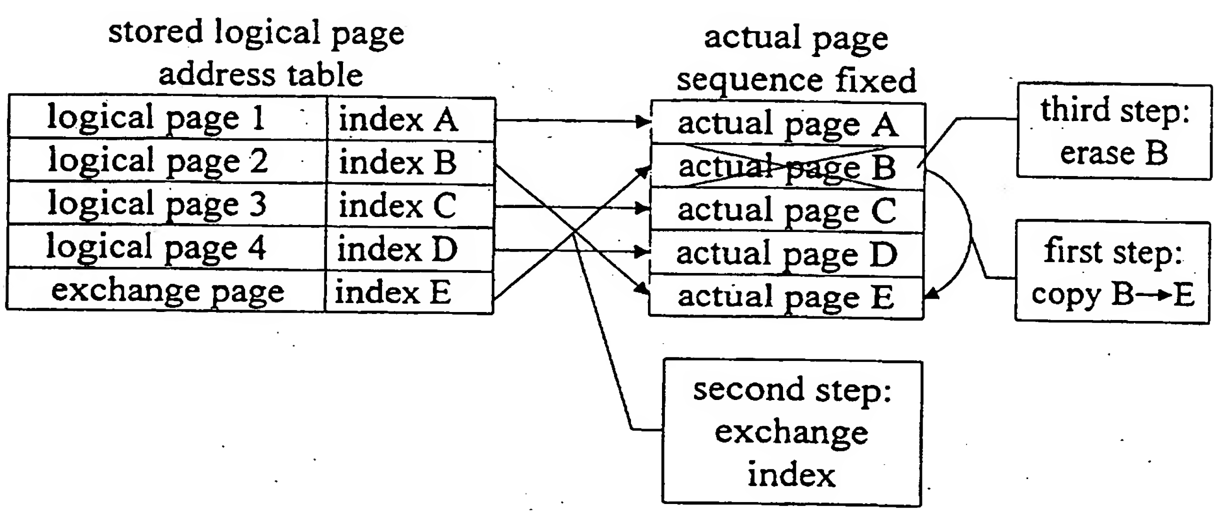
stored logical page address table
logical page 1
index A
logical page 2
index B
logical page 3
index C
logical page 4
index D
exchange page
index E
actual page sequence fixed
actual page A
~~actual page~~ B
actual page C
actual page D
actual page E
third step: erase B
first step: copy B→E
second step: exchange index

Fig. 5 B

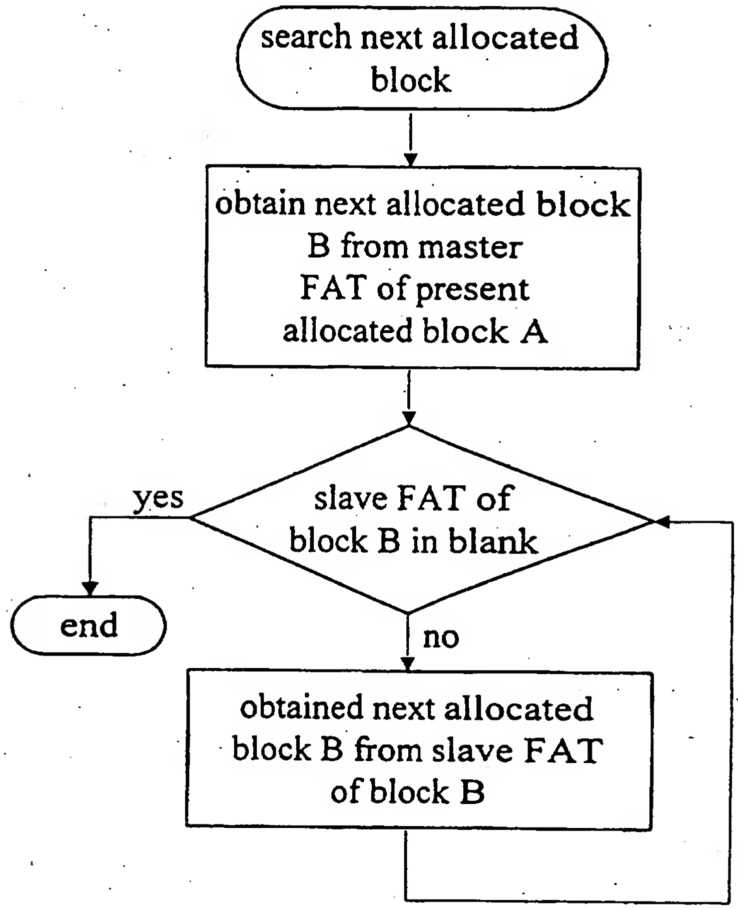
search next allocated block
obtain next allocated block B from master FAT of present allocated block A
slave FAT of block B in blank
yes
end
no
obtained next allocated block B from slave FAT of block B

Fig. 6

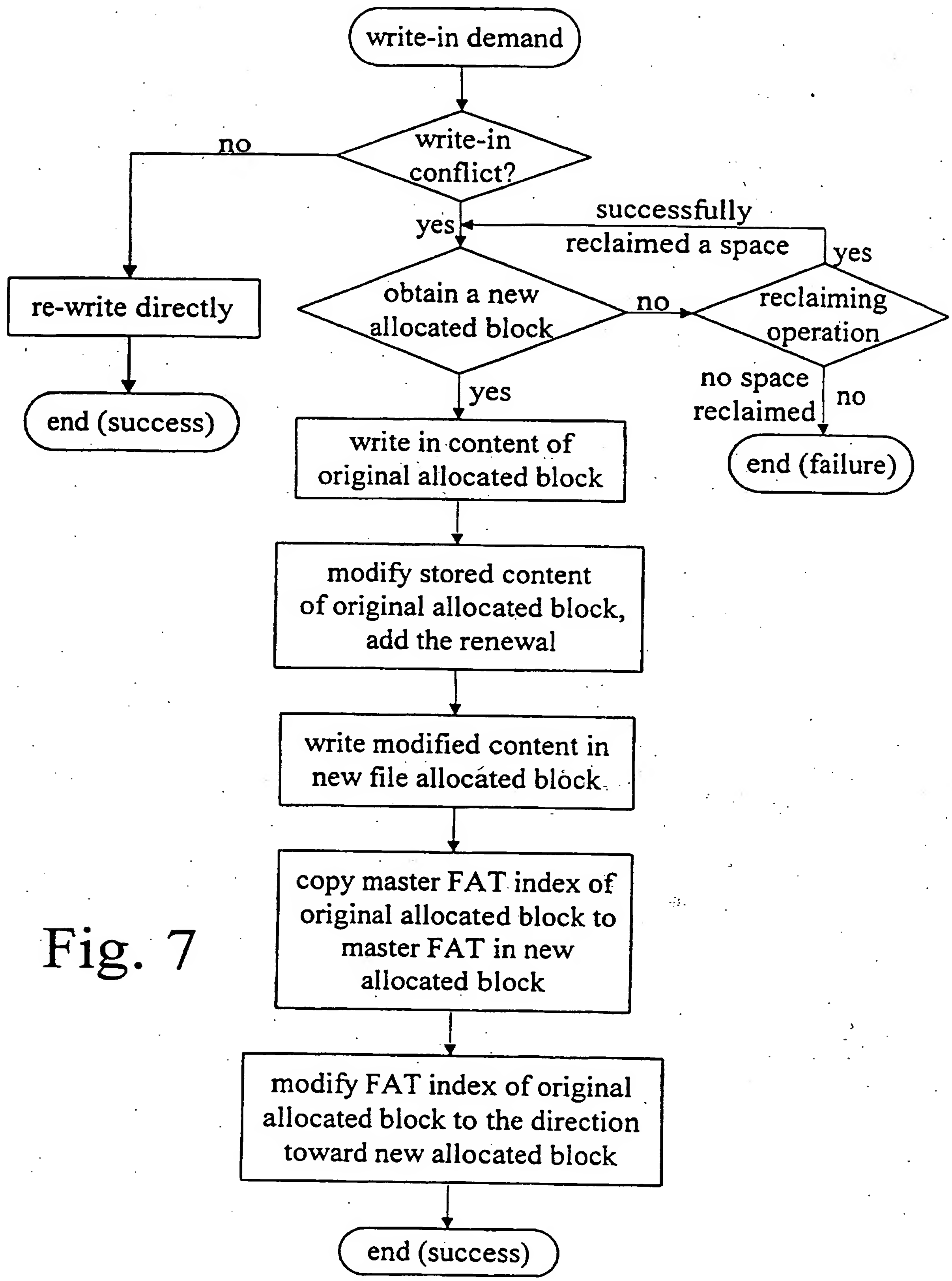
write-in demand
write-in conflict?
no
yes
re-write directly
end (success)
successfully reclaimed a space
yes
obtain a new allocated block
no
reclaiming operation
no space reclaimed
no
end (failure)
yes
write in content of original allocated block
modify stored content of original allocated block, add the renewal
write modified content in new file allocated block
copy master FAT index of original allocated block to master FAT in new allocated block
modify FAT index of original allocated block to the direction toward new allocated block
end (success)

Fig. 7

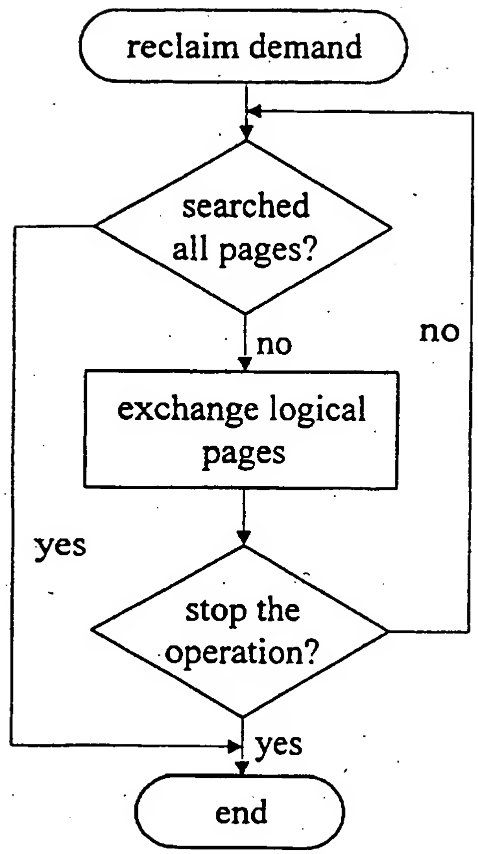
reclaim demand
searched all pages?
no
exchange logical pages
no
yes
stop the operation?
yes
end

Fig. 8

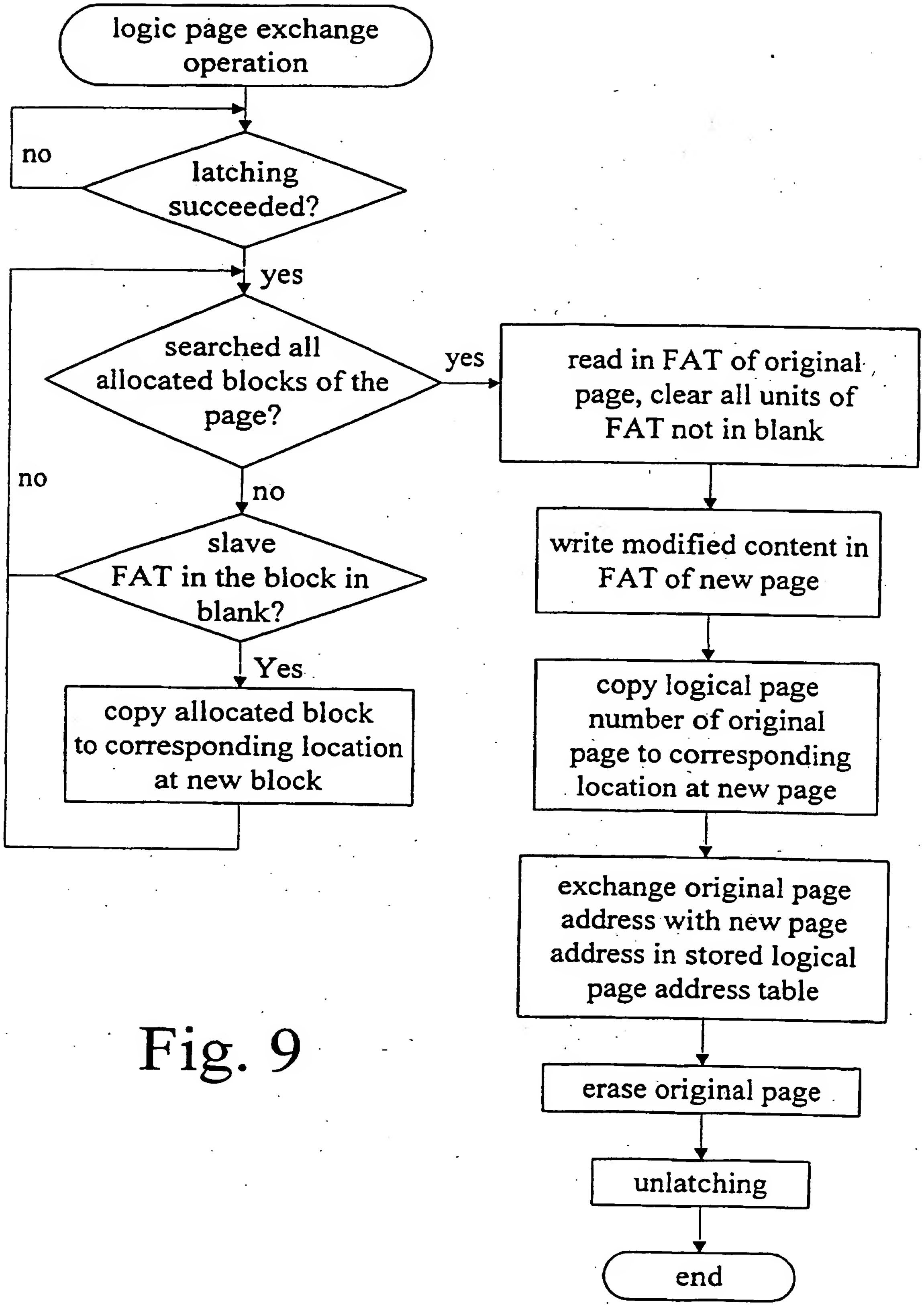
logic page exchange operation
latching succeeded?
no
yes
searched all allocated blocks of the page?
yes
no
slave FAT in the block in blank?
no
Yes
copy allocated block to corresponding location at new block
read in FAT of original page, clear all units of FAT not in blank
write modified content in FAT of new page
copy logical page number of original page to corresponding location at new page
exchange original page address with new page address in stored logical page address table
erase original page
unlatching
end

Fig. 9

FLASH MEMORY ARCHITECTURE AND ITS OPERATING METHOD

The present invention relates to a flash memory architecture which enables the performance of the flash memory to be closer to an EEPROM (electrically erasable programmable read only memory) under normal operation, and improved to 436 times (if file allocated block is 256 bytes) or 870 times (if file allocated block is 128 bytes), so that file loss due to power failure, flash memory replacement or other accidents during file writing is eliminated, and file modification record is reserved within a certain length of time, enabling the user to search the required file or to resume lost file during tracing or file resume operation.

When rewriting FAT (file allocated table) of the file itself during normal operation of a flash memory, the whole file block must be erased before rewriting. For example, when modifying a particular item in a 16-bit FAT, i.e., when rewriting a string of characters in for example TOSHIBA TC58F400 flash memory, the FAT rewriting procedure includes the steps of:

A. reading in the block to be erased (needs to read in 16K file) ;

B. erasing the old block (requires at least 1500ms);

C. rewriting a string of characters (a word) in the memory; and

D. writing back the file in the memory (writing in 16K file

requires 16μs*16000-256ms).

As stated above, rewriting one work uses 1750ms, and the file is at a dangerous status within this rewriting period of time. As indicated, the aforesaid conventional method has drawbacks as follows:

1. Because the whole file block must be erased when writing FAT and the file itself during normal system operation, the performance of the whole flash memory architecture and operating method is greatly lowered, and the service life of the flash memory is affected.
2. In case power failure, flash memory replacement or an accident occurs during rewriting, it causes a big loss of file, or a damage to the FAT in the flash memory.

In view of the drawbacks of the aforesaid conventional flash memory operating method, the present invention provides a new design of flash memory architecture and its operating method. One object of the present invention is to provide a new flash memory architecture which enables the performance of the flash memory to be closer to an EEPROM (electrically erasable programmable read only memory) under normal operation, and improved to 436 times (if file allocated block is 256 bytes) or 870 times (if file allocated block is 128 bytes). Another object of the

present invention is to provide a new flash memory architecture which eliminates a file loss due to power failure, flash memory replacement or other accidents during a file writing operation, so as to improve the stability of the system. Still another object of the present invention is to provide a flash memory architecture which enables file modification record to be reserved within a certain length of time (subject to residual space of the flash memory), so that the user can search the required file, or resume lost file. Still another object of the present invention is to provide a new flash memory operating method which combines renewal content and the content in the original allocated block together, then write the combined content in the new allocated block, and then change the index direction to the new allocated block, enabling the new allocated block to substitute for the original allocated block, so as to eliminate the operation of erasing the whole file allocated block and the possibility of a file loss due to power failure, system down, or a replacement of the flash memory. Still another object of the present invention is to provide a new flash memory operating method which keeps file modification record to be reserved within a certain length of time, enabling the user to search the required file or to resume lost file during tracing or file resume operation.

The invention will now be described further by way of example with reference to the accompanying drawings in which:

Figure 1A illustrates the application of the flash memory operating method according to the present invention (I).

Figure 1B illustrates the application of the flash memory operating method according to the present invention (II).

Figure 1C illustrates the application of the flash memory operating method according to the present invention (III).

Figure 2A is a FAT (file allocated table) architecture according to the present invention.

Figure 2B is an application example of the FAT architecture according to the present invention.

Figure 3 explains the operating principle of the master-slave FAT (file allocated table) according to the present invention.

Figure 4 explains the operating principle of the abandoned allocated block reclaiming work according to the present invention.

Figure 5A illustrates initialization of logical pages according to the present invention.

Figure 5B explains the operation of logical page exchange according to the present invention.

Figure 6 is a flow chart explaining the file writing operation in the flash memory according to the present invention.

Figure 7 is a flow chart explaining the file rewriting or FAT rewriting operation according to the present invention.

Figure 8 is a flow chart explaining the abandoned allocated

block reclaiming operation according to the present invention.

Figure 9 is a flow chart explaining the logical page exchanging operation according to the present invention.

The present invention provides a new design of flash memory architecture and its operating method. As illustrated in Figures 1A, 1B and 1C, when rewriting the flash memory, an index is set up corresponding to allocated blocks at the flash memory, and allocated blocks are selectively erased instead of erasing the whole flash memory, i.e., when rewriting the flash memory, the content to be re-written and the content in the original allocated block are composed and then written into a new allocated block, and at the same time the index is aimed at the new allocated block, enabling the new allocated block to be substituted for the old allocated block.

The flash memory architecture comprises master-slave FAT (file allocated table) and logical pages. The logical pages are defined subject to the number of erasable blocks at the flash memory. The whole block of the flash memory is divided into a series of logical pages, each logical page is identified by a particular page number. One of the logical pages is an exchange page (see Figures 5A and 5B).

Master-Slave FAT comprises FATs (file allocated tables),

namely, Master FAT **FAT1** for data recording, and Slave FAT **FAT2** for use during correction of Master FAT **FAT1**, enabling FAT to be distributed to every logical page so as to reduce erasing operation risk (see Figure 2A). Figure 2B shows Master-Slave FAT set up in a TC58F400 flash memory (of which the erasable block is 16K) from TOSHIBA according to the aforesaid method.

Referring to Figure 3, when a CPU works with a flash memory having Master-Slave FAT and logical pages, tree structure is used to substitute for the single chain structure of Master FAT and Slave FAT, and every block (containing three allocated blocks) of the tree structure represents one FAT, in which the first allocated block is the file allocated block represented by the FAT itself, and does not exist in actual file architecture and operating method, and is contained at the location of the FAT itself; the second allocated block is the next FAT indicated by the FAT, namely, Master FAT; the third allocated block is Slave FAT.

For further understanding of the present invention, the present invention is explained by way of example as follows:

When renewing the content of a 0x81 file allocated block, the CPU (not shown) obtains a new file allocated block (for example 0x900) from Slave FAT in the flash memory, then writes down the new file allocated block (0x900) in Slave FAT at the 0x801 file allocated block, and then the usable content in the

0x801 file allocated block and the content to be written are copied to the new file allocated block, enabling Master FAT in the new file allocated block to be aimed at the location indicated by Master FAT of the 0x801 file allocated block.

If slave FAT of file allocated block is in blank when the CPU analyzing FAT in the flash memory, it proceeds analyzing procedure subject to table keys of master FAT. On the contrary, if slave FAT of file allocated block is not in blank, slave FAT oriented new file allocated block is used to substitute for the original block. This condition forms a key table itself. As shown in Figure 3, 0x802→0x901→0x902 is a modified key table obtained after several modifications.

As indicated above, if any space available, the CPU needs not to proceed with an erasing operation as conventionally did, but simply to move a small piece of data file. This design saves much time in renewing content, and allows a rewriting operation to be performed without erasing a big piece of data file.

If cannot obtain a new file allocated block when analyzing FAT, the CPU reclaims abandoned file allocated blocks from file allocated table in the flash memory (see 0x801, 0x802, 0x901 in Figure 4). Generally, the CPU reclaims abandoned file allocated blocks under the following two situations. One situation is that the CPU does no work. At this situation, the CPU starts abandoned

file allocated block reclaiming procedure to perform the reclaiming work. The other situation is that the distribution of a new file allocated block fails. When the distribution of a new file allocated block fails, the abandoned file allocated block reclaiming procedure is started.

The work of reclaiming abandoned file allocated block is done by the CPU by means of scanning the whole FATs (including master FAT and slave FAT) in the flash memory. If the slave FAT of one particular file allocated block is found not in blank, the index of the master FAT of the file allocated block is modified and aimed at the correct file allocated block, and all the file allocated block in the slave FAT that is found not in blank is erased (see Figure 4).

Referring to Figure 5A, when logical pages match with respective master-slave file tables to make an exchange, the CPU initializes all logical pages of the matched flash memory, and makes a respective group of numbers arranged in an order subject to logical page numbers. The content is the actual address of the respective page. The exchange page is put behind the logical pages. The logical page number and the serial number of the respective allocated block in the logical page are mixed. Therefore, no matter where the logical page is, we can know the location of the allocated block to be operated in the logical pages through the FAT

and the location of the logical page in which the allocated block is allocated. Thus, the main work of the CPU is to exchange pages but not to write modified file in the original location of the file when rewriting the flash memory.

Referring to Figure 5B, new file in the memory is copied to the exchange page, and then the address of the exchange page is exchanged with the erased page. Thus, the erased page becomes a new exchange page, and the logical page sign is erased when erasing an exchange page.

Further, the flash memory operation method of the present invention matches with the architecture of the aforesaid new flash memory architecture. The operational flow of the method is outlined hereinafter.

Referring to Figure 7, when renewing a file in the flash memory, the CPU finds out the allocated block from the set master-slave FAT in the flash memory, and judges if any write-in conflict would occur in the found allocated block, and then directly write the renewal file in the allocated block if the judgment result is negative, and then the work is ended.

After the end of the work, the CPU keeps searching other allocated blocks. When a new allocated block is obtained, the CPU firstly reads in the content from the original FAT, then modifies the original FAT, and then adds the new content and writes it with

modified content in the new allocated block, and then copies the index of the master-slave FAT in the old allocated block to the master-slave FAT in the new allocated block, and then modifies the index direction of the slave FAT in the original allocated block, and thus the re-write work is done.

With respect to the action of inquiring next allocated block, please refer to Figure 6. The first step is to search the next allocated block subject to the master FAT index at the allocated block of the old file. If the slave FAT been searched is not in blank, the allocated block aimed by the slave FAT is used to substitute for the original allocated block. This procedure is repeated again and again till a blank allocated block is obtained. When an allocated block in which the slave FAT is in blank is obtained, the search action is ended.

Further, if no any blank allocated block is obtained after all allocated blocks have been searched, the CPU starts reclaiming abandoned allocated blocks. If no any abandoned allocated block is reclaimed, an information of failure appears, and the work is ended. On the contrary, if an abandoned allocated block is reclaimed, the reclaimed abandoned allocated block is changed to a new allocated block.

The operation of reclaiming an abandoned allocated block is outlined hereinafter with reference to Figure 8. At first, it

searches all logical pages in the flash memory, and ends the searching after all logical pages have been searched. If not all logical pages have been searched, it proceeds to the step of exchanging the logical pages. During logical page exchanging, the CPU judges if to provide a stop demand to the flash memory subject to actual operation condition, so as not to be occupied for long. If the judgment result is positive, the work is stopped. If negative, it keeps searching logical pages in the flash memory, and performing logical page exchange operation.

With respect to logical page exchanging operation, please refer to Figure 9. At first, the CPU judges if latching succeeds or not. If negative, the CPU keeps proceeding the work of latching till the work is done. Thus, the processing procedure will not be interrupted, all allocated blocks in the logical pages can be continuously searched. When all allocated blocks in the logical pages are continuously searched,, the CPU judges if all logical pages have been searched or not. If the judgment result is positive, the CPU reads in the FAT of the original logical page, then clears the file from the slave FAT, and then writes the modified content in the FAT of the new logical page, and then copies the page number of the original logical page to the location corresponding to the new logical page, and then exchanges the page address of the original logical page with the page address of the new logical page,

and then erases the file set in the original logical page, and then ends the work after unlatching. If the judgment result is negative, the CPU keeps judging if the slave FAT of searched allocated block is in blank or not. If the judgment result is positive, the CPU copies the allocated block to location corresponding to the new logical page. The work is done after all allocated blocks have been searched.

As indicated above, the invention of "Flash memory architecture and its operating method" eliminates the drawbacks of conventional techniques, and achieves improvement in actual practice.

CLAIMS

1. A flash memory architecture comprising:

a series of logical pages divided from the whole block of a flash memory, said logical pages each having a respective logical page number for identification, one of said logical pages being an exchange page;

a master-slave FAT (file allocated table), said master-slave FAT including a master FAT for recording file and a slave FAT set up in the flash memory for use in modifying master FAT.

2. The flash memory architecture of claim 1 wherein said master-slave FAT is distributed in every of said logical pages.

3. A flash memory operating method comprising the step of:

(a) enabling a CPU to search allocated blocks in the master-slave FAT set in a flash memory;

(b) enabling the CPU to read in the content of the allocated block that is stored with the original file when the CPU obtained a new allocated block, and then enabling the CPU to modify the content of the old allocated block and to add the portion to be renewed;

(c) enabling the CPU to write in the new allocated block the content modified subject to the aforesaid steps (a) and (b);

(d) enabling the CPU to copy the index of the master FAT in the original allocated block to the master FAT in the new

allocated block;

(e) enabling the CPU to modify the index direction of the slave FAT in the original allocated block toward the new allocated block.

4. The flash memory operating method of claim 3 wherein the CPU judges if a write-in conflict occurs in the searched allocated block during allocated block searching operation, and then directly writes renewal file in the allocated block if the judgment result is negative, or keeps searching new allocated block if the judgment result is positive.

5. The flash memory operating method of claim 3 wherein when searching a new allocated block, the CPU obtains a new allocated block from the master FAT index at the allocated block, and uses the slave FAT oriented allocated block to substitute for the original allocated block if the slave FAT of the searched allocated block is not in blank.

6. The flash memory operating method of claim 3 wherein the CPU starts an abandoned allocated block reclaiming operation when obtaining no new allocated block, and then ends the work if no abandoned allocated block is obtained, otherwise changes the reclaimed allocated block thus obtained to a new allocated block.

7. The flash memory operating method of claim 6 wherein the abandoned allocated block reclaiming operation comprises the

steps of:

(a) searching logical pages in the flash memory, and ending the searching after all logical pages have been searched; and

(b) if not all logical pages have been searched, proceeding to the step of exchanging the logical pages in which the CPU judges if to provide a stop demand to the flash memory subject to actual operation condition, so as not to be occupied for long, and then stops the work if the judgment result is positive, or keeps searching logical pages in the flash memory and performing logical page exchange operation if the judgment result is negative.

8. The flash memory operating method of claim 7 wherein the logical page exchange operation comprises the steps of:

(a) judging if latching succeeds or not;

(b) continuously proceeding the work of latching till the work is done if latching is not succeeded, so that the processing procedure will not be interrupted, all allocated blocks in the logical pages can be continuously searched, and when all allocated blocks in the logical pages are continuously searched, the CPU judges if all logical pages have been searched or not, then the CPU reads in the FAT of the original logical page if the judgment result is positive, and then clears the file from the slave FAT, and then writes the modified content in the FAT of the new logical page, and then copies the page number of the old logical page to the

location corresponding to the new logical page, and then exchanges the page address of the original logical page with the page address of the new logical page, and then erases the file set in the original logical page, and then ends the work after unlatching, and the CPU keeps judging if the slave FAT of searched allocated block is in blank or not if the judgment result is negative; and

(c) the CPU copies the allocated block to location corresponding to the new logical page if latching succeeds, and the work is done after all allocated blocks have been searched.

9. Flash memory architecture substantially as herein described with reference to and as illustrated in the accompanying drawings.

Application No: GB 9908869.2
Claims searched: 1-9
Examiner: Melanie Gee
Date of search: 17 November 1999

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): G4A (AFL, AMX, ANR, ANV)

Int Cl (Ed.6): G06F 12/02, 12/06, 12/10; G11C 16/02, 16/06

Other: Online: WPI, EPODOC, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
X, E	GB 2333620 A	(INTEL), see whole document.	1 & 3 at least
X	GB 2291990 A	(MEMORY CORPORATION), see whole document.	1 & 3 at least
X	WO 98/44420 A1	(LEXAR MEDIA), see especially page 7 line 7 - page 9 line 2.	1 & 3 at least
A	US 5682497 A	(ROBINSON), see whole document.	
X	US 5566314 A	(DE MARCO et al.), see whole document.	1 & 3 at least
X	JP090161491 A	(FUJITSU), see abstract.	1 & 3 at least.

17

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
		E	Patent document published on or after, but with priority date earlier than, the filing date of this application.
&	Member of the same patent family		

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☑ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.